

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-260826

(43)Date of publication of application : 29.09.1998

(51)Int.Cl. G06F 9/06
G06F 9/46
G06F 12/00
G06F 12/00
G06F 12/00

(21)Application number : 09-062754

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
NEC CORP

(22)Date of filing : 17.03.1997

(72)Inventor : SUNAGA HIROSHI
NAKAMURA RYOICHI
YAMADA TETSUYASU
OCHI KENICHI

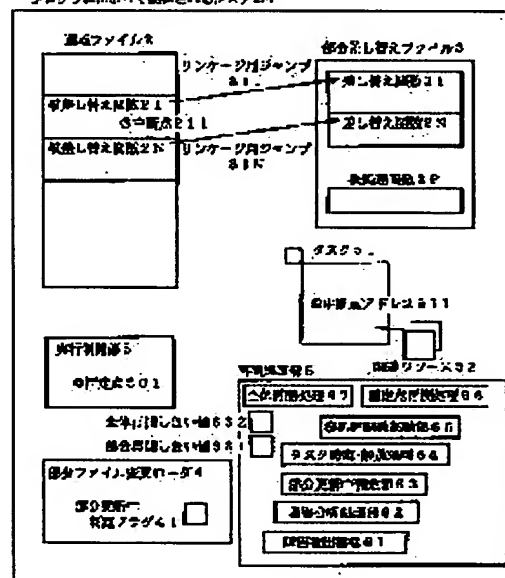
(54) PARTIAL FILE UPDATING METHOD DURING SYSTEM OPERATION

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent the operation interruption of an entire system with receiving only a partial effect even though a conflict occurs because of an interrupting task and it might be a system failure at the time of partially changing an operation file and also to make the system perform normal operation recovery even when a fault is repeated in spite of execution of the main processing.

SOLUTION: A failure detecting mechanism 61 detects a failure, a failure analysis processing part 62 analyzes if the failure is caused by a software factor, a task specification and release processing part 64 specifies a program parallel execution processing unit that generates the failure when it is caused by the software factor and resets the task 51 in which a conflict occurs and at the same time, a post-processing function starting part 65 specifies a resource about 52 about the task and starts a function that initializes. A fixed point restart processing part 66 resumes the operation of a system at the fixed point 501 of an execution controlling part 5.

プログラムによって制御されるシステム1



LEGAL STATUS

[Date of request for examination] 17.05.2001

[Date of sending the examiner's decision of rejection] 04.03.2005

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-260826

(43) 公開日 平成10年(1998) 9月29日

(51) Int.Cl.⁸
G 0 6 F 9/06
9/46
12/00
識別記号
5 4 0
3 3 0
5 1 8
5 3 1
5 3 5

F I
G 0 6 F 9/06
9/46
12/00
5 4 0 F
3 3 0 C
5 1 8 A
5 3 1 R
5 3 5 Z

審査請求 未請求 請求項の数4 O L (全 10 頁)

(21) 出願番号 特願平9-62754

(22) 出願日 平成9年(1997) 3月17日

(71) 出願人 000004226

日本電信電話株式会社
東京都新宿区西新宿三丁目19番2号

(71) 出願人 000004237

日本電気株式会社
東京都港区芝五丁目7番1号

(72) 発明者 須永 宏

東京都新宿区西新宿三丁目19番2号 日本
電信電話株式会社内

(72) 発明者 中村 亮一

東京都新宿区西新宿三丁目19番2号 日本
電信電話株式会社内

(74) 代理人 弁理士 磯村 雅俊 (外1名)

最終頁に続く

(54) 【発明の名称】 システム運転中部分ファイル更新方法

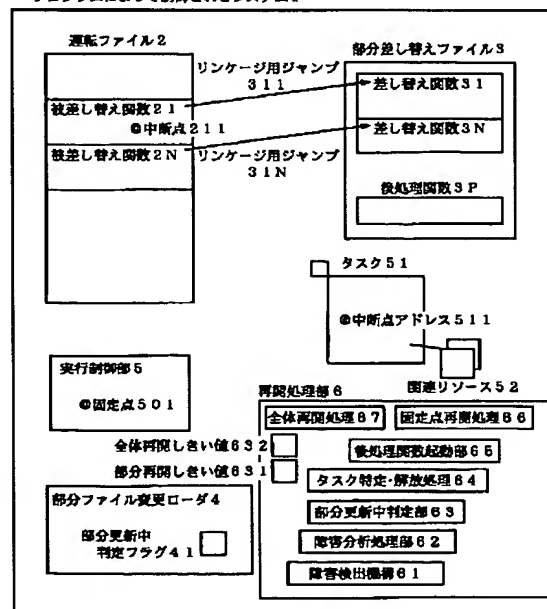
(57) 【要約】

【課題】 運転ファイルを部分変更する際に、中断タスクのために矛盾が発生しシステム障害となるところ、部分的影響のみでシステム全体の運転中断を防止するとともに、本処理を行っても障害を繰り返す場合でも、システムの正常な運転回復を行わせる。

【解決手段】 障害検出機構61で障害を検出し、障害分析処理部62でその障害がソフトウェア要因によるものかを分析し、そうであれば、タスク特定・解放処理部64で障害を発生させたプログラム並列実行処理単位を特定して、矛盾が発生したタスク51をリセットすると同時に、後処理関数起動部65がそのタスクに関するリソース52を特定して初期設定する関数を起動する。固定点再開処理部66は、実行制御部5の固定点501からシステムの運転を再開させる。

本発明の原理ブロック図

プログラムによって制御されるシステム1



【特許請求の範囲】

【請求項1】 運転ファイルに対して部分差し替えファイルをロード・リンクしてプログラムの変更を行う通信・情報処理システムのファイル更新方法において、

部分ファイル更新ローダが上記運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更している際に、該部分差し替えファイルが組み込まれてから予め定めた監視時間内に発生した障害を検出し、

該障害がソフトウェア要因によるものか否かを分析し、分析の結果、ソフトウェア要因であれば、該障害を発生させたプログラム並列実行処理単位を特定して、特定された該実行単位単位を消去し、

プログラム実行制御の特定の点から処理を再開させることを特徴とするシステム運転中部分ファイル更新方法。

【請求項2】 請求項1に記載のシステム運転中部分ファイル更新方法において、

前記部分ファイル更新ローダが運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更する際に、該部分差し替えファイルを運転中のシステムに投入すると同時に、該部分差し替えファイルの盛り込みにより発生し得るソフトウェア障害に関するリソースを解放するために記述した後処理プログラムを該システムに割り付け、

運転中のファイルを部分的に差し替えている期間中に、ソフトウェア障害が発生した場合には、該ソフトウェア障害を引き起こしたプログラム並列実行処理単位を消去した後、上記後処理プログラムを起動させて該処理プログラムによるリソースを解放し、

該リソース解放処理後にプログラム実行制御の特定の点から処理を再開させることを特徴とするシステム運転中部分ファイル更新方法。

【請求項3】 請求項2に記載のシステム運転中部分ファイル更新方法において、

前記運転中のファイルを部分的に差し替えている期間中に発生するソフトウェア要因の障害発生回数のしきい値を指定する手段を設け、

システムは上記期間中の障害発生回数を計数して、上記指定手段のしきい値に達した時点で該当する部分差し替えファイルを取り外すことを特徴とするシステム運転中部分ファイル更新方法。

【請求項4】 請求項3に記載のシステム運転中部分ファイル更新方法において、

前記前記運転中のファイルを部分的に差し替えている期間中に発生するソフトウェア要因の障害発生回数について少なくとも2種類のしきい値を指定する手段を設け、システムは上記期間中の障害発生回数を計数して、先ず最初のしきい値の回数に達した時点で、該当する部分差し替えファイルを取り外し、

該部分差し替えファイルの取り外し後、再度障害が発生して次のしきい値の回数に達した時点で、該システム全

体を再開させることを特徴とするシステム運転中部分ファイル更新方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、プログラムにより運転の制御が行われるシステムのファイル管理方法に関し、特に運転ファイルを部分的に変更する際に、複数箇所の関数を修正しても中断タスクがあるために矛盾が発生してシステム障害になるところ、部分的な影響のみでシステム全体の運転の妨げをなくし、システム障害の可能性を低減することができるようなシステム運転中部分ファイル更新方法に関する。

【0002】

【従来の技術】 従来より、プログラムにより運転されるシステムでは、プログラムの瑕疵の修正、機能の追加等によりファイルを部分的に差し替えてプログラムの変更を行っている。この場合に、部分差し替えファイル投入直前でタスクが中断していると、部分差し替えファイルが割り付けられた直後にそのタスクが中断点から再開した場合、本タスクが格納しているスタック情報と部分差し替えファイルの新プログラム処理との間で処理矛盾が生じて、システム異常になる可能性がある。そのため、システム異常を回避しながら運転中ファイルの変更を行うことが重要となる。図13は、従来のシステム構成例を示すブロック図である。図13において、1はプログラムにより制御されるシステム、2はシステム建設時、あるいはファイル全体更新時に設定された原本の運転ファイル、21～2Nは運転ファイルを構成するプログラム単位（ここでは、関数と呼ぶ）のうち、置換される対象のもの、211は被差し替え関数の中断点、3は部分差し替えファイル、31、3Nは部分差し替えファイルを構成する差し替え対象の関数、4は差し替え関数をメモリ上に割り付けて、原本ファイルの被差し替え対象関数の代りにリンクし直す機能を備えた部分ファイル変更ローダ、41は部分更新中判定フラグ、5は実行制御部、501は固定点、51は実行制御部により提供されるタスクと呼ばれる並列実行単位、6は再開処理部、61は障害検出部、62は障害分析処理部、67は全体再開処理部である。なお、上記障害検出部61は、未実装番地やプロテクトのかかった命令エリアを走行したりすると、ハードウェア的に割り込みを起こさせるプロセッサ内のハードウェア機構であるが、実行制御部5、タスク51、および再開処理部6のその他機能部は、メモリ上に格納され、事象が発生した際にプロセッサにより駆動されるプログラムで構成される。勿論、ファームウェア等で組み込むこともできる。また、運転ファイル2および部分差し替えファイル3は、メモリ上のデータを格納するためのエリアを備えたファイルである。

【0003】 現在運転中のファイル2に瑕疵がある場合、あるいは運転中ファイル2に機能を追加したい場合

には、部分差し替えファイル3を投入していく。しかし、部分差し替えファイル3を投入する前にタスク51が被差し替え関数21内で中断していると（中断点211で中断）、ローダ4により新しい差し替え関数31、3Nに対してリンクが取られる。この中断タスク51は、新しい差し替え関数31の処理は経験していないが、差し替え処理の終了後に実行を再開すると、新しい差し替え関数3Nを走行することとなる。この場合、タスク51が新しい差し替え関数31の処理無しで関数3Nを走行することになるため、関数3Nが関数31の処理結果を前提としているような処理を持っていれば、中断再開後、タスク51の動作が異常になる可能性が高い。すなわち、差し替え関数3Nを実行するには、差し替え関数31の処理を経由していないと矛盾が生じる論理になっている。この矛盾は、軽微な誤動作ではなく、未実装番地へアクセスしたり、データ部にアクセスしたり、あるいは誤ったアドレスのデータを破壊する等のソフトウェア障害を引き起こすため、ソフトウェアの全初期設定によりシステムの運転を回復させる必要があった。すなわち、ソフトウェア障害が発生すると、障害検出部61により検出されてシステム全体再開処理67が起動され、タスク等の全消去の後にシステムの運転が再開される。しかし、このような再開処理が起こると、運転中のシステムが中断し、提供中のサービスが提供できなくなるという問題がある。

【0004】

【発明が解決しようとする課題】このように、従来のシステムにおいては、運転中ファイルを部分差し替える場合に、差し替えプログラムの内容が正しくても、変更前に中断していたタスクが差し替え処理完了直後に処理再開した時点で処理矛盾を起こし、結局はソフトウェア障害となって、安定した運用ができなくなる場合があった。そこで、本発明の目的は、このような従来の課題を解決し、運転中の原本ファイルに対して部分差し替えファイルをロード・リンクする直前に残っていた中断タスクが、新しく盛り込まれた差し替えファイルにより未実装番地アクセス、データ部走行、データ破壊等のソフトウェア障害を引き起こす可能性があっても、ソフトウェア障害を起こしたタスクのみを消去するのみで、浮きリソースの発生を防ぎ、中断タスクでソフトウェア障害を起こしたものの以外のサービスを継続させることが可能なシステム運転中部分ファイル更新方法を提供することにある。また、本発明の他の目的は、このような部分的な再開処理で、システムの正常な運転が回復できない場合には、システムの正常な運転回復のためのガードとなるようなシステム運転中部分ファイル更新方法を提供することにある。

【0005】

【課題を解決するための手段】上記目的を達成するため、本発明のシステム運転中部分ファイル更新方法で

は、運転中のファイルを部分的に差し替える部分ファイル更新ローダが、運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更する場合に、障害検出部が差し替えファイルが組み込まれてから一定の監視時間内に発生した障害を検出し、障害分析処理部がその障害がソフトウェア要因によるものか否かを分析し、もしソフトウェア要因である場合には、タスク特定・解放処理部がその障害を発生させたプログラム並列実行処理単位を特定した後に消去し、固定点再開処理部が実行制御の特定の点から処理を再開する。さらに、障害を引き起こしたプログラム並列実行処理単位を消去した後に、後処理関数起動部が後処理プログラムを起動させて、プログラムによるリソース解放処理後に固定点再開処理部が実行制御の特定の点から処理を再開する。さらに、部分的な再開処理でシステムの正常な運転が回復できない場合には、障害の繰返しをカウントして、しきい値を越えた場合にシステム全体を再開させることにより、システムの運転回復を保証する。

【0006】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。図1は、本発明の一実施例を示すシステムのブロック構成図である。図1に示すように、本発明においては、従来例の図13と同一部分を備えており、図13と異なる部分は、再開処理部6の構成を追加した点と、部分差し替えファイル3中に後処理関数3Pを設けた点と、タスク51に関連リソース52を設けた点である。図13と同一のものには、同一の符号を付して示している。すなわち、1はプログラムにより制御されるシステム、2はシステム建設時あるいはファイル全体更新時に設定された原本の運転ファイル、21～2Nは運転ファイル2を構成する関数のうち置換される対象のもの、3は部分差し替えファイル、31、3Nは部分差し替えファイルを構成する差し替え対象の関数、4は差し替え関数をメモリ上に割り付けて、原本ファイルの被差し替え対象関数の代わりにリンクし直す機能とそれをアンロード（取り外し）する機能を備えた部分ファイル変更ローダ、41は部分更新中判定フラグ、5は実行制御部、501はタスク起動制御の開始点である固定点、51は実行制御部により提供されるタスクと呼ばれる並列実行単位、52は関連リソース、6はシステム再開処理部である。システム再開処理部6には、障害検出機構61、ソフトウェア障害か否かを判断する分析処理部62、システムが部分ファイル変更中か否かを確認する部分ファイル変更中判定部63、罹障タスクを特定してこれを解放させるためのタスク特定・解放処理64、後処理関数3Pを起動する後処理関数起動部65、実行制御部5の固定点501に戻り、実行制御を続行する固定点再開処理66、システムの全体的な再開を実施する全体再開処理67、および後処理関数を起動した後も、罹障を繰返すか否かを判定する部分再開しきい値63

1、およびその罹障がさらに繰り返すか否かを判定するための全体再開しきい値632が設けられている。なお、上記障害検出機構61のみは、ハードウェア的に割り込みを起こさせる機構であって、ハードウェア構成である。また、運転ファイル2および差し替えファイル3は、メモリのデータを格納するエリアを備えたファイルであり、それ以外の機能部は全てメモリ上に格納され、事象が発生した際にプロセッサにより駆動されるプログラムである。勿論、論理をファームウェア等で組み込んだ装置（プロセッサ）とすることも可能である。

【0007】図1においては、運転中の原本ファイル2に対して、被差し替え対象の関数21、2Nを差し替える関数31と3Nに、差し替えファイル31と3Nによって引き起こされ得るソフトウェア障害に関わるタスクに関連するリソースを消去するための処理を記述した関数3Pを添付する。さらに、再開処理部6では、障害検出部61がローダ4によるメモリ上への割り付け後に発生した障害を検出し、分析部62がソフトウェア障害であるか否かを判定し、部分ファイル変更中判定部63がシステムの部分ファイル変更中であるか否かを確認し、タスク特定・解放処理防64が罹障タスクを特定して解放し、後処理関数起動部65が後処理関数3Pを起動し、全体再開処理部67がシステムの全体的な再開を実施する。これにより、ローダ4が差し替え処理を実行する直前に中断していたタスクが、差し替えファイル31と3Nがリンクされたことによって矛盾を生じ、ソフトウェア障害を引き起こした時に、それがソフトウェア障害であって、かつ罹障タスク51を特定できた場合に、そのタスクを強制消去する。そして、そのタスクに関連していたリソースを消去するための後処理関数3Pを起動し、その関数に記述されている通りのリソース消去処理を行った上で実行制御5の固定点501に復帰する。部分ファイル差し替えの直前に中断していたタスクに関連する部分以外については、全く影響を与えることなく運転中のファイルを部分変更することが可能である。さらに、後処理関数を起動させた後に罹障が繰り返されるか否かを部分再開しきい値631によって判定し、もし、しきい値オーバーの場合には、タスク特定・解放処理部64によりそのタスクを強制消去する。同時に、ローダ4を起動して部分差し替えファイル31、3Nをアンロードすることにより部分差し替え前の状態に戻す。また、その後も罹障を繰り返す場合には、その回数を計数して、全体再開しきい値632を越えた場合には、システム全体再開処理67を起動し、システムの運転を回復させる。

【0008】図2～図4は、本発明の動作原理を示す説明図である。なお、図1と同一のものには同一の符号を付して示している。そして、図2、図3、図4の順序で状態が変化していく様子を示している。図2は、運転中ファイルに対して部分差し替えファイルを投入する前

で、実行制御内でタスクが中断している状態を示している。すなわち、実行制御部5により制御されるタスク51が、中断点アドレス511として被差し替え関数21内の中断点211のアドレスを格納している。図3では、部分差し替えファイル3がシステムに割り付けられた直後にそのタスク51が起動して、タスク51の処理を中断点アドレス511から再開するが、タスク中断点511の前方と後方に差し替えられた関数が置かれており、そのタスク51は差し替えられた関数の処理のうち、後ろの半分しか通らないために、差し替え関数内で処理矛盾が発生した状態を示している。すなわち、中断からの再改後、差し替え関数3Nを動くと、ソフトウェア異常が発生する。これにより、障害検出機構61が起動される。また、図4では、障害検出機構61により検出し、障害分析処理部63で分析の結果、発生した障害がソフトウェア障害であり、罹障タスクを特定して、そのタスクのリセットとそのタスクに係るリソースを差し替えファイルと同時に入力した後処理関数によって消去して、他の部分には影響がなく部分ファイル差し替えが完了していることを示している。

【0009】図5～図12は、本発明の一実施例を示す詳細動作の説明図である。前述の動作原理を、実際の動作により詳細に説明する。図5に示すように、部分差し替えファイル3が割り付けられる前で、被差し替え関数21の中でタスク51が中断しているものとする。タスク51の実行は実行制御部5により制御されるが、タスク51は中断点アドレス511として、被差し替え関数21内の中断点211のアドレスを格納している。この時、図6に示すように、ローダ4により差し替えファイル3内の関数31、3Nが割り付けられ、被差し替え関数21、2Nとの間でリンケージがなされる（リンケージ用ジャンプ311、31N）。このようにして、部分差し替えファイル3が投入されると、ローダ4の機能により差し替え関数31と3N、および後処理関数3Pがシステム上に割り付けられる。なお、リンケージ処理直前に中断していたタスク51は、リンケージ後までそのまま残る。次に、図7に示すように、ローダ4は、部分差し替え関数31と3Nが割り付けを終了すると、実行制御部5に制御を渡し、実行制御部5はタスク51を中断点から起動させて中断点アドレス211から処理を再開させようとする。しかし、タスク51は該中断点にて就寝していたため、差し替え関数31の処理を経験せず差し替え関数3Nの処理を走行しようとする。そのため、本来、関数31と3Nの組み合わせにおいて、1つの正当な処理がなされるところが実際にはそのようにならないために、処理矛盾が発生して、システム障害となる（障害を×で示す）。

【0010】次に、図8においては、障害検出機構61が障害を検出し、障害分析処理部62でソフトウェア要因の障害であることを判定すると、部分更新中判定部6

3により部分ファイル更新ロード4内の部分更新中フラグ41を判定し、かつ部分再開しきい値631を判定する。部分更新中でしきい値以内であれば、タスク特定解放処理部64において、罹障を引き起こしたタスクを特定し、そのタスク51を消去する（消去を×で示す）。タスク消去に引き続いて、図9に示すように、後処理関数起動部65により後処理関数3Pを起動し、そのタスクに関連していたリソース511を消去する（消去を×で示す）。後処理関数3Pは、差し替え関数31、3Nをコーディングするプログラマが、中断タスクが存在して引き起こす可能性のある障害時に関連リソースを特定して解放する処理を記述したものである。この関数は、この時のみ起動されるものであるが、その起動の仕方は実装に依存する。本実施例では、関数名に特定の名称またはサフィックスを付加してその関数を識別し、割り付け位置はその名称をアドレスを対応させるデータとして所持しているものとする。なお、図8および図9において、実装によっては、他の処理が割り込まないように、実行制御部5に対して割り込みマスクを設定する。

【0011】図10は、関連リソース消去の後に、固定点再開処理部66により実行制御部5の固定点501から通常の運転を再開させる流れを示したものである。すなわち、関連リソース52の消去から戻ると、固定点再開処理部66を経由して、実行制御部5の固定点501からシステムの運転を再開させる。図11は、更新部分しきい値をオーバーした場合の処理の説明図である。後処理関数起動部65を起動してタスクやリソースを消去しても、障害を繰り返して、部分再開しきい値631をオーバーした場合には、新たに割り付けた部分差し替えファイル31、3Nに瑕疵があり、タスクの強制終了では罹障を回復できない場合であるとみなす。この場合には、タスクを解放するとともに、部分再開ロード4（部分ファイル変更ロードと同じ）により部分差し替えファイル31、3Nの割り付けを元に戻し（アンロードする）、リンケージを戻して固定点再開処理部66により実行制御部5の固定点501から通常の運転を再開させる処理を行う。なお、上記しきい値は、一定の条件、例えば一定時間という条件でリセットし、無関係な要因発生によりしきい値オーバーする確率を下けている。図12は、部分差し替えファイルに戻しても回復できないような障害に陥った場合、および今回の部分ファイル差し替えと関係ない要因でソフトウェア障害が起こっている場合か否かを判断する動作を示している。例えば、一定時間内にソフトウェア障害が全体再開しきい値632をオーバーしたならば、全体再開処理67によりシステム全体再開を行うことにより、システムを回復できる確率を高めることができる。

【0012】

【発明の効果】以上説明したように、本発明によれば、プログラムにより制御されるシステムで、運転ファイル

を部分的に変更する際に、複数箇所の関数を修正しても中断タスクがあるために矛盾が発生してシステム障害になるところ、矛盾が発生したタスクをリセットするとともに、そのタスクに関するリソースを特定して初期設定する関数を起動することにより、部分的な影響のみでシステム全体の運転の妨げをなくすることができる。また、上記の処理を行っても罹障を繰り返す場合には、部分差し替えファイルに戻して固定点再開させるか、あるいはシステム全体を再開させることにより、システムの正常な運転回復のためのガードを行うことができる。その結果、本発明によれば、運転中のファイルを部分更新する場合のシステム障害を防止することができるので、安定してシステムを運転させることができる。

【図面の簡単な説明】

【図1】本発明によるシステム運転中部分ファイル更新の動作原理を示す図である。

【図2】本発明によるシステム運転中部分ファイル更新の概略動作の説明図（部分差し替えファイル投入前）である。

【図3】同じく概略動作の説明図（部分差し替え関数内で処理矛盾が発生した状態）である。

【図4】同じく概略動作の説明図（罹障タスクのリセットと、リソースを消去する処理）である。

【図5】本発明の一実施例を示すシステム運転中部分ファイル更新の詳細動作の説明図（部分差し替えファイルの割り付け前）である。

【図6】同じく詳細動作の説明図（差し替え関数および後処理関数の割り付け処理）である。

【図7】同じく詳細動作の説明図（処理矛盾の発生とシステム障害）である。

【図8】同じく詳細動作の説明図（部分更新中でしきい値以内のため、タスクの消去を行う）である。

【図9】同じく詳細動作の説明図（関連リソースの消去）である。

【図10】同じく詳細動作の説明図（固定点再開処理）である。

【図11】同じく詳細動作の説明図（部分しきい値をオーバーした場合）である。

【図12】同じく詳細動作の説明図（部分差し替えファイルに戻しても回復できない障害の場合）である。

【図13】従来のシステム運転中部分ファイル更新の動作説明図である。

【符号の説明】

1…プログラムにより制御されるシステム、2…運転ファイル、21～2N…運転ファイルを構成するプログラム単位（関数）（差し替えられる対象の被差し替え関数）、211…タスク中断点、3…部分差し替えファイル、4…部分ファイル変更ロード、3P…差し替え関数を入れたことにより発生し得るソフトウェア障害に関連するリソースを消去する後処理を記述した後処理関数、

31～3N…差し替えファイル内で、差し替え用の部分差し替え関数、41…部分更新中判定フラグ、5…タスクの実行を司る実行制御部、51…プログラム実行単位タスク、501…実行制御の起点となる固定点、511…タスク51が関数中で中断しているアドレスを示す中断点アドレス、52…タスクが関連するリソース、61…障害検出機構、6…障害発生した場合の回復処理を司る再開処理部、62…検出した障害がソフトウェア障害か否かを判定する障害分析処理部、63…障害発生時、部分更新中フラグ41をチェックして部分更新中か否かを判定する部分更新中判定部、631…障害が繰り返

10

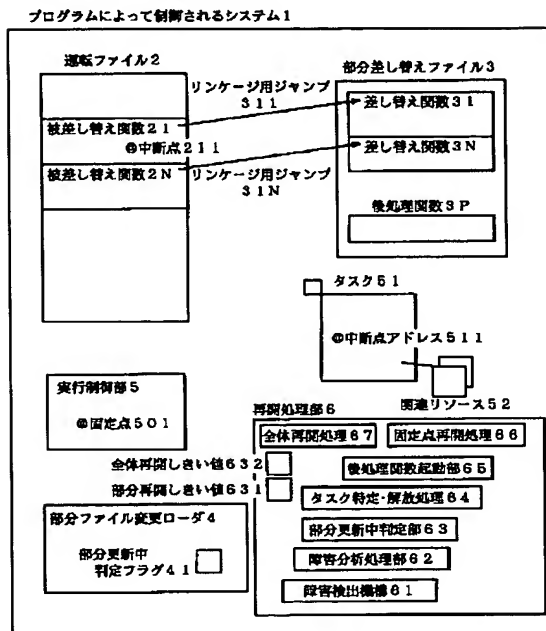
し、部分差し替えファイルをアンロードするか否かを判定する部分再開しきい値、632…障害が繰り返し、システム全体再開を起動するか否かを判定する全体再開しきい値、64…障害タスクを特定し、解放するタスク特定・解放処理部、

65…差し替えファイル3の投入と同時に割り付けられる後処理関数3Pを起動する後処理関数起動部、66…実行制御の固定点から再開するための処理を行う固定点再開処理部、67…システム全体の再開を処理する全体再開処理部。

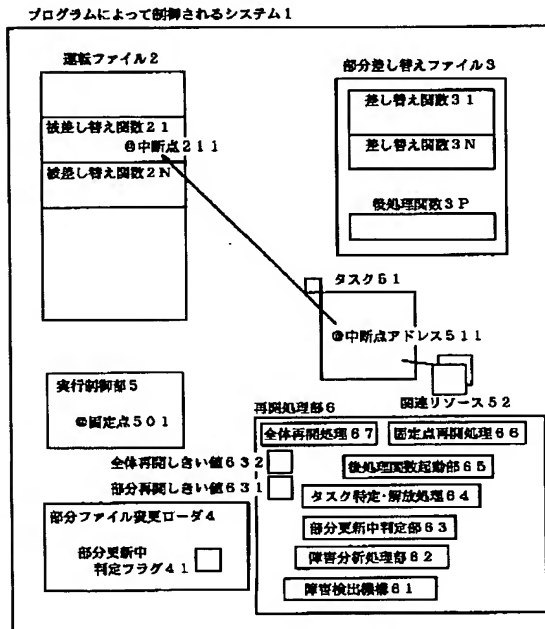
【図1】

【図2】

本発明の原理ブロック図

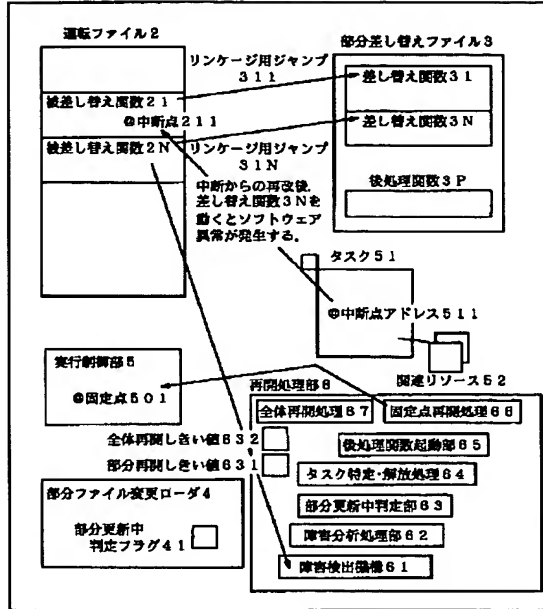


本発明の原理動作説明図



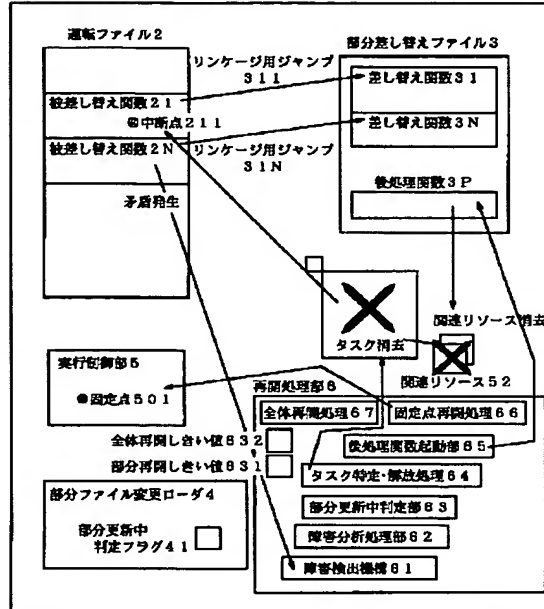
【図3】

プログラムによって制御されるシステム1



【図4】

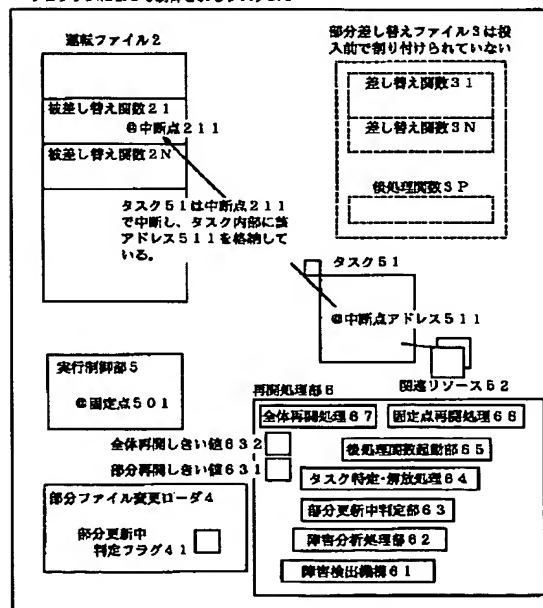
プログラムによって制御されるシステム1



【図5】

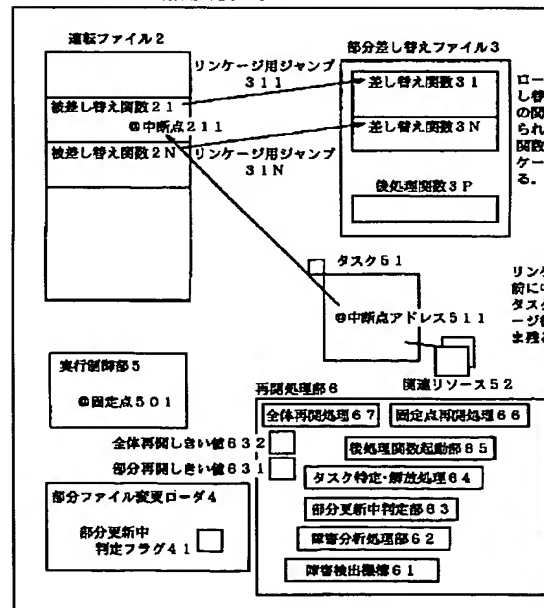
本発明の動作の一例を示す図

プログラムによって制御されるシステム1

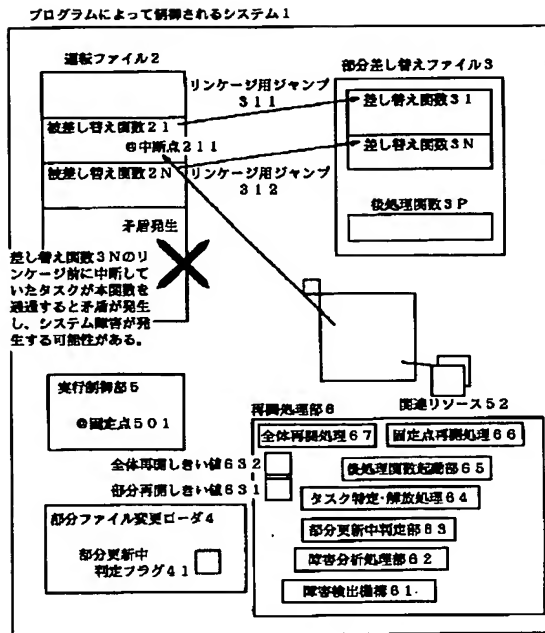


【図6】

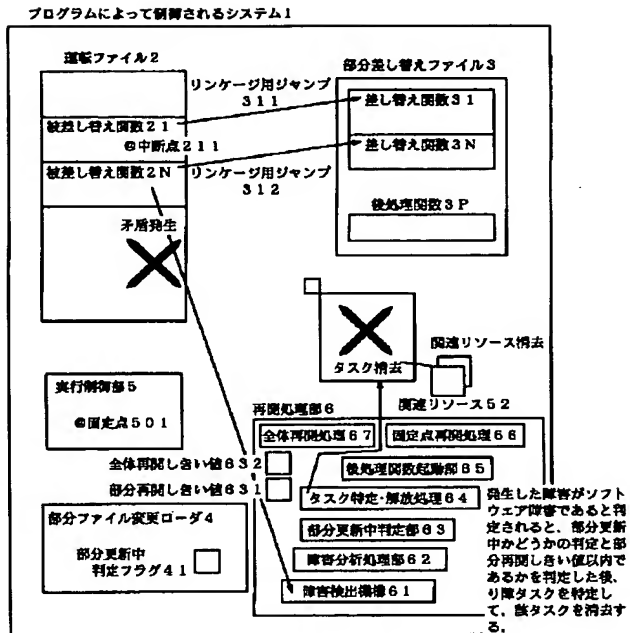
プログラムによって制御されるシステム1



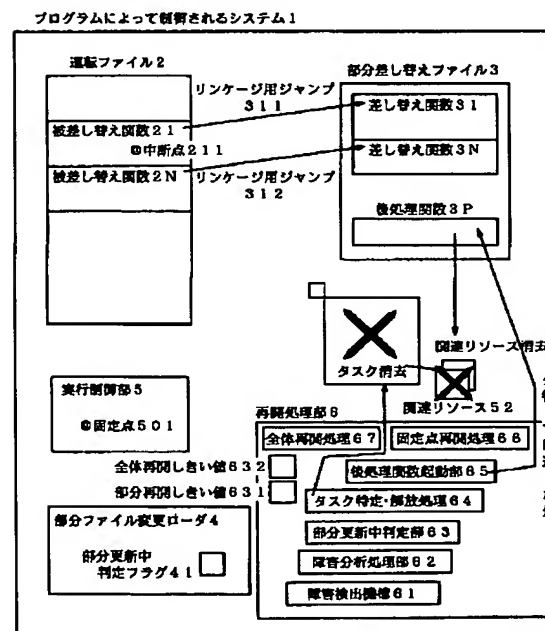
【図7】



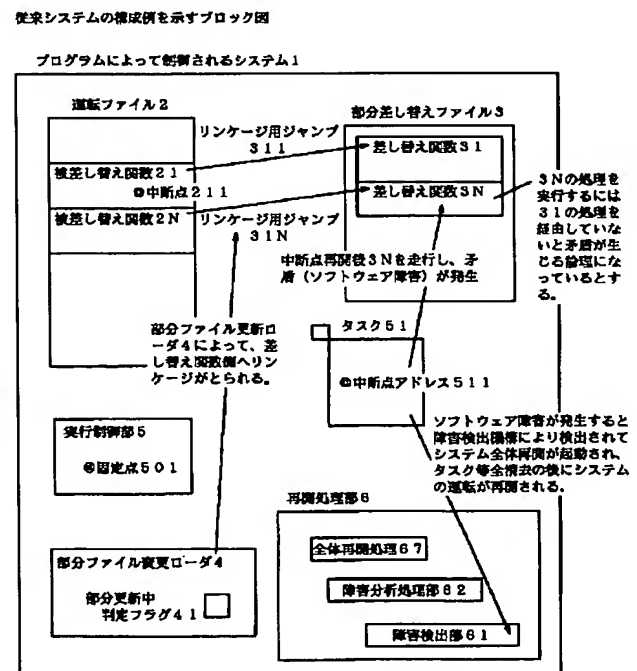
【図8】



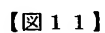
【図9】



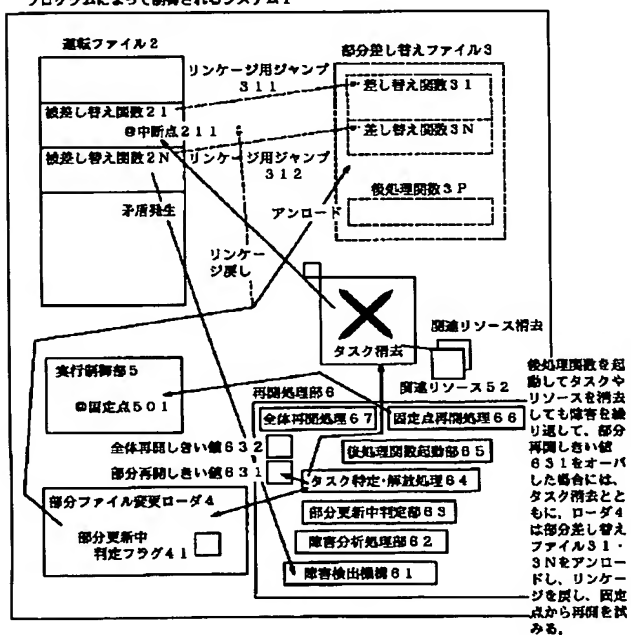
【図13】



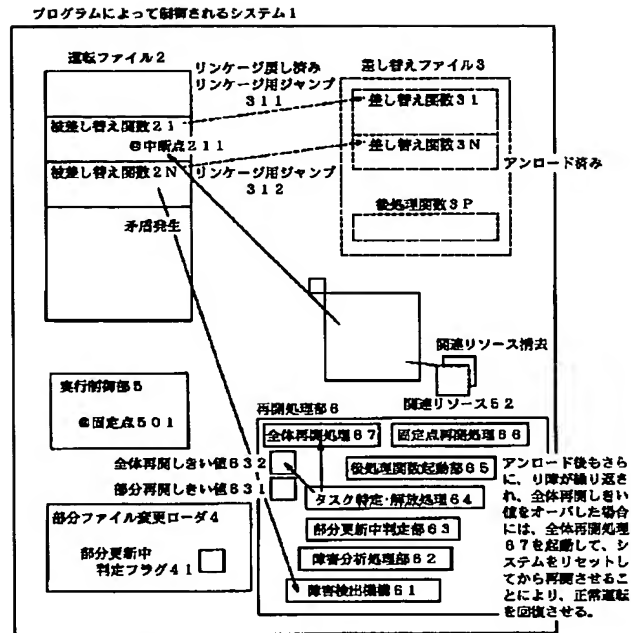
プログラムによって制御されるシステム1



プログラムによって制御されるシステム 1



【図12】



フロントページの続き

(72)発明者 山田 哲靖

東京都新宿区西新宿三丁目19番2号 日本
電信電話株式会社内

(72)発明者 越智 憲一

東京都港区芝五丁目7番1号 日本電気株
式会社内